REMARKS

The Specification has been amended for clarification purposes only, and thus, does not present new matter. Claims 1, 2, 4, 6, 8, 9, 11, 13, and 15 have been amended. Claims 16-19 have been added and claims 7 and 14 have been cancelled. Thus, claims 1-6, 8-13 and 15-19 are currently pending in the case. Further examination and reconsideration of the presently claimed application is hereby respectfully requested.

Objection to the Specification

The Specification was objected for an informality. In particular, the Office Action suggests that on page 41, the statement in the paragraph labeled by the number 2 is in the form of an if-then statement where the then or assumed then is missing. To expedite prosecution, the Specification has been amended in a manner that addresses those concerns expressed in the Office Action. Accordingly, removal of the objection to the Specification is respectfully requested.

Section 103 Rejections

Claims 1-15 were rejected under 35 U.S.C. § 103(a) as being unpatentable over a publication provided by MageLang Institute, entitled *Swing Short Course, Part 1* (hereinafter "MageLang") in view of a publication provided by Sun Microsystems, entitled *Java Platform 1.2 Beta 4 API Specification: Class LookAndFeel* (hereinafter "JavaSPEC") and in further view of a publication provided by JavaOne, entitled "Java Foundation Classes (a.k.a. "Swing") Component Architecture" (hereinafter "JavaOne").

To establish a *prima facie* obviousness of a claimed invention, all claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974), MPEP 2143.03. Obviousness cannot be established by combining or modifying the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion or incentive to do so. *In re Bond*, 910 F. 2d 81, 834, 15 USPQ2d 1566, 1568 (Fed. Cir. 1990). The cited art does not teach or suggest all limitations of the currently pending claims, some distinctive limitations of which are set forth in more detail below.

None of the cited art teaches or suggests a system of software components (claim 1), a computer-readable storage device (claim 15) or a method (claim 8) for establishing color inheritance between a parent object and a child object, where the color inheritance is established in the following order: (i) if a background color of the child object is declared, the child object is displayed with the declared background color, (ii) if the background color of the child object is not declared, and a background color is globally defined for the system of components, the child object is displayed with the globally defined background color, which is independent of the operating system, and (iii) if the background color of the child object is not declared, and the background color is not globally defined for the system of components, the child object is displayed with the background color of the parent. Amended independent claim 8 states in part:

> A method for color inheritance between a parent object and a child object... wherein the method is adapted to display the child object in accordance with the following color inheritance order: if a background color of the child object is declared, displaying the child object with the declared background color; if the background color of the child object is not declared, and a background color is globally defined for the system of components, displaying the child object with the globally defined background color; and if the background color of the child object is not declared, and the background color setting is not globally defined for the system of components, displaying the child object with the background color of the parent.

Amended independent claims 1 and 15 recite similar limitations. Support for the amendments made to claims 1, 8 and 15 may be found in cancelled claims 7 and 14, in the Specification, for example, on page 41, lines 1-20, and in Fig. 19b.

Generally speaking, the Specification provides a system, method and computer-readable storage device for overcoming one of the many problems that occurs when programmers attempt to mix AWT and Swing components within a Java application program. In one embodiment, the presently claimed case provides a system of software components – referred to as "AWTSwing" components – which enable a child object to be displayed with a background color that is either (i) explicitly declared for that child object, (ii) assigned by global "look and feel" Swing settings (which are independent of the operating system executing the software components), or (iii) inherited from a parent of the child object. The method for establishing the inheritance of background color for the child object is purposefully performed in the order presented above. *See*, e.g., Specification, page 40, line 8 to page 42, line 7.

With regard to present claim 8, the Examiner suggests that "MageLang teaches a method of color inheritance (see page 7 under the JButton heading) between a system of components (see page 4)...to display a parent (parent window) and child object (the button) (see page 7 under the heading JButton)."

(Office Action, pages 4-5). However, the Examiner admits that MageLang fails to teach "a middle step of checking for a globally defined color, and if defined, setting the background to the defined color" (Office Action, page 5). The Examiner provides similar suggestions for the presently claimed system of software components (claim 1) and the presently claimed computer-readable storage device (claim 15) on pages 2-3 and 5-6 of the Office Action. In response to the Examiner's suggestions, the Applicant agrees that there is no teaching or suggestion within MageLang for the so-called "middle step" of present claims 1, 8 and 15, and further asserts that there is also no teaching or suggestion within MageLang for the last limitation of those claims.

The MageLang reference provides a brief overview of Swing objects that may be included within a class of objects commonly known as the "JComponents" (*See*, e.g., MageLang, pages 1-35). Referring to page 4 of MageLang, the "JComboBox", "JLabel", "JButton", and "JPasswordField" objects are all Swing objects that descend from the JComponent class. Page 4 of MageLang also shows how the Swing objects of the JComponent class may be the "child objects" of one or more "parent objects". For example, the "JButton" object is shown as a child object of the "AbstractButton" parent object, which in turn is a child object of the "JComponent" class. As pointed out by the Examiner, these child objects may inherit their background color from a parent object, if a background color is not explicitly declared for a particular child object. In most cases, the background color can be explicitly declared for a child object by using the "setBackground" method from the Java "Component" class. However, and as noted in the Specification, the "setBackground" method "automatically returns the background color of the component's parent when there is no explicit declaration for the component." (Specification, page 42, lines 4-7). Since the method fails to determine whether or not a background color has been globally defined, the method cannot be used to teach or suggest any claim limitation that depends on the determination of a globally defined background color (e.g., the second and third limitations of claims 1 and 8). As a consequence, the MageLang reference cannot be relied upon to teach or suggest the presently claimed method of color inheritance, as recited in claims 1, 8, and 15.

The Examiner suggests that teaching can be found within the JavaOne and JavaSPEC references for establishing a globally defined background color. For example, the Examiner states, "JavaOne teaches, on attached pages 7 and 8, building default tables to store default colors, fonts, icons and borders for components, and that all subsequent objects will use the new values." (Office Action, page 3, 5 and 6). The Examiner also states, "[i]t is further taught in JavaSPEC, on page 3, initializing the colors of the background to the default color from the defaults table." (Office Action, page 3, 5 and 6). The Applicant

concedes that default colors, fonts, icons and borders may be stored in a Swing defaults table to establish the global "look and feel" settings for Swing child objects. The use of Swing default tables is commonly known in the art. However, the mere mention of Swing default tables does not and cannot provide the presently claimed method of color inheritance, as recited in claims 1, 8 and 15.

JavaOne and JavaSPEC both mention the Swing defaults table. For example, the JavaSPEC reference provides a brief overview of the Swing "LookAndFeel" class, including various methods for applying the look and feel settings that may be stored within a Swing defaults table (*See*, e.g., JavaSPEC, pages 1-7). In particular, the JavaSPEC reference indicates that the "installColorsAndFont" method of the "LookAndFeel" class may be used for initializing a component's (e.g., a child object's) foreground, background and font properties with the values obtained from a currently set defaults table. As such, the "installColorsAndFont" method may be used for applying a default background color to a child object. However, the JavaSPEC reference fails to provide a means for determining whether or not "look and feel" settings exist within a current defaults table, without automatically applying those settings to the child object. For example, the JavaSPEC reference does not suggest that the default background color may not be applied if a background color is explicitly declared for a child object. In other words, even though the JavaOne and JavaSPEC references may disclose the existence of Swing default tables, they do not disclose the presently claimed method by which the default tables may be used. Therefore, the JavaOne and JavaSPEC references cannot be relied upon to teach or suggest the presently claimed method of color inheritance, as recited in claims 1, 8, and 15.

The Examiner suggests that the "sequence of hierarchical inheritance is further taught by Bogdan, in column 1, line 65 through column 2, line 32, column 5, line 32, and column 7, lines 65 through column 8, line 14." (Office Action, pages 3 and 6). The Applicant respectfully disagrees, for at least the reasons set forth in more detail below.

Bogdan discloses a "system that supports a uniform three-dimensional rendering of components on a display screen by allowing a new component to inherit the shading properties of a parent component." (Bogdan, Abstract). Bogdan, however, does not teach or suggest the presently claimed method of color inheritance. Instead, Bogdan provides a substantially different method of color inheritance, as shown in Fig. 3 of Bogdan. Bogdan specifically states that "if a shading property is not defined in a given child component at the time the component is created, that child component can search its parent component hierarchy for inheritable shading properties. Absent any other source, shading properties and other

properties can be obtained from available system properties. However, using inherited properties ensures that each component in a component family hierarchy is consistent in its display characteristics." (Bogdan, column 2, lines 6-14, emphasis added; *see*, Fig. 3). Thus, the method of Bogdan differs from the presently claimed method of color inheritance, in one respect, by attempting to use inheritable shading properties (from a parent component) before "system-wide properties" may be used.

The "system-wide properties" of Bogdan are not independent of the operating system (*see*, e.g., Bogdan, column 1, lines 41-52). As such, the "system-wide properties" of Bogdan cannot be considered equivalent to a "globally defined background color", which is currently defined in present claims 1, 8 and 15 as independent of the operating system.

There is no motivation within Bogdan to modify the teachings of Bogdan to provide the presently claimed globally defined (and operating system independent) background color. However, such a case may be considered for the sake of argument. As described in more detail below, even if the "system-wide properties" of Bogdan were somehow modified to be independent of the operating system, the "system-wide properties" of Bogdan could not be used in the manner set forth in present claims 1, 8 and 15.

For example, Bogdan emphasizes "that only a root or top-level component will take its shading face color from the system property settings if no other color is expressly stated." (Bogdan, column 7, lines 35-37). Since the "system-wide properties" of Bogdan are only applied to root or top-level components, the "system-wide properties" of Bogdan cannot be applied to child objects, as taught in the presently claimed case. Furthermore, the "system-wide properties" of Bogdan cannot be applied in the order set forth in the present claims, since Bogdan teaches away from using system-wide properties before using inheritable properties. As noted above, Bogdan utilizes system-wide properties only if a background color cannot be obtained from any other source (*See*, Bogdan, column 2, lines 6-14). Bogdan appears to consider the use of system-wide properties a disadvantage of prior art systems (*See*, e.g., Bogdan, column 1, lines 40-47). Therefore, even if the "system-wide properties" of Bogdan were somehow modified to: (1) be independent of the operating system, and (2) be applied to child objects instead of top-level components, the teachings of Bogdan would still provide no motivation for using system-wide properties before using inheritable properties, because Bogdan specifically teaches away from doing so. It is noted that a *prima facie* case of obviousness may be rebutted by showing that the art, in any material respect, teaches away from the claimed invention. *In re Geisler*, 116 F.3d 1465, 1471, 43 USPQ2d 1362, 1366 (Fed. Cir. 1997); MPEP 2144.05 (III). As a consequence, Applicants strongly believe that the teachings of

Bogdan cannot be relied upon to provide teaching or suggestion for the presently claimed method of color inheritance, as recited in claims 1, 8, and 15.

For at least the reasons set forth above, MageLang, JavaOne, JavaSPEC and Bogdan each fail to teach or suggest the presently claimed method of color inheritance. However, the Examiner suggests that "it would have been obvious to one of ordinary skill in the art, having the teachings of MageLang, JavaSPEC, JavaOne, and Bogdan before him at the time the invention was made to modify the JButton system of MageLang to include the use of a default table... because default tables allow for a system to have specific settings, which make all windows uniform." (*See*, e.g., Office Action, page 3). As described below, the JButton system of MageLang, the default tables of JavaOne and JavaSPEC, and the inheritance sequence of Bogdan cannot be combined to provide the presently claimed method of color inheritance.

It is improper to combine references where the references teach away from their combination. *In re Grasselli*, 713 F.2d 731, 743, 218 USPQ 769, 779 (Fed. Cir. 1983). MPEP 2145 (X)(D)(2). Since Bogdan teaches away from the claimed invention, the inheritance sequence of Bogdan cannot be combined with the remaining cited art to teach or suggest all limitations of the presently claimed case. Furthermore, the remaining cited art cannot be combined or modified in such a manner that teaches or suggests the presently claimed method of color inheritance.

Any methods of color inheritance that may be disclosed by MageLang, JavaOne and JavaSPEC are bounded by predefined methodologies, or in other words, sequences of programming code that have been established and accepted as Swing methods for setting the background color. One simply cannot combine the methods of MageLang, JavaOne and JavaSPEC without significantly modifying their normal programming code sequence. For example, one simply cannot combine the "setBackground" method (inherently disclosed by MageLang) with the "installColorsAndFont" method (disclosed by JavaOne and JavaSPEC) to provide the presently claimed method of color inheritance without making any modifications to those methods. MageLang, JavaOne and JavaSPEC provide no indication that these methods could be modified to provide the presently claimed method of color inheritance. Therefore, Applicants assert that the MageLang, JavaOne and JavaSPEC references cannot be combined or modified to teach or suggest all limitations of present claims 1, 8 and 15.

For at least the reasons set forth above, none of the cited art, either separately or in combination provides motivation to teach or suggest all limitations of present claims 1, 8, and 15. Therefore, claims 1, 8, and 15, as well as claims dependent therefrom, are patentably distinct over the cited art. Claims 7 and 14 have been canceled rendering rejection thereto moot. Accordingly, removal of this rejection is respectfully requested.

Patentability of the Added Claims

The present amendment adds claims 16-19. Claims 16-19 are dependent from claim 1, and thus, are patentably distinct over the cited art for at least the same reasons as claim 1. Accordingly, allowance of added claims 16-19 is respectfully requested.

CONCLUSION

This response constitutes a complete response to all issues raised in the Office Action mailed January 15, 2004. In view of the remarks traversing rejections, Applicants assert that pending claims 1-19 are in condition for allowance. If the Examiner has any questions, comments, or suggestions, the undersigned attorney earnestly requests a telephone conference.

No fees are required for filing this amendment; however, the Commissioner is authorized to charge any additional fees which may be required, or credit any overpayment, to Conley Rose, P.C. Deposit Account No. 03-2769/5468-07700.

Respectfully submitted,

Kevin L. Daffer
Reg. No. 34,146
Attorney for Applicant(s)

Conley Rose, P.C.
P.O. Box 684908
Austin, TX 78768-4908
Ph: (512) 476-1400
Date: April 14, 2004
JMF